ASSIGNMENT OF SEMANTIC TAGS TO PHRASES FOR GRAMMAR GENERATION

The present invention relates to the field of automated language understanding for dialogue applications.

Automatic dialogue systems and telephone based machine enquiry systems are nowadays widely spread for providing information, as e.g. train or flight timetables or receiving enquiries from a user, as e.g. bank transactions or travel bookings. The crucial task of an automatic dialogue system consists of the extraction of necessary information for the dialogue system from a user input, which is typically provided by speech.

The extraction of information from speech can be divided into the two steps of speech recognition on the one hand side and mapping of recognized speech to semantic meanings on the other hand side. The speech recognition step provides a transformation of the speech received from a user in a form that can be machine processed. It is then of essential importance, that the recognized speech is interpreted by the automatic dialogue system in the correct way. Therefore, an assignment or a mapping of recognized speech to a semantic meaning has to be performed by the automatic dialogue system. For example for a train timetable dialogue system the enquiry "I need a connection from Hamburg to Munich", the two cities "Hamburg" and "Munich" have to be properly identified as origin and destination of the train travel.

Essential fragments of the above sentence "from Hamburg" or "to Munich" have to be extracted and to be understood by the automatic dialogue system to the extent, that the phrase "from Hamburg" is mapped to the origin semantic tag whereas the phrase "to Munich" is mapped to the destination semantic tag. When all semantic tags like origin, destination, time, date, or other travel specifications are mapped to phrases of the user enquiry, the dialogue system can perform a required action.

The assignment of mapping of recognized phrases to semantic tags is typically provided by some kind of grammar. A grammar contains rules defining the mapping of semantic tags to the phrases. Such rule based grammars have been the most investigated subject of research in the field of natural language understanding and are often incorporated in actual dialogue systems. An example of an automatic dialogue system as well as a general description of automatic dialogue systems is given in the paper "H. Aust, M. Oerder, F. Seide, V. Steinbiss; the Philips Automatic Train Timetable Information System, Speech Communication 17 (1995) 249-262".

Since an automatic dialogue system is typically designated to a distinct purpose, as e.g. a timetable information or an enquiry processing system, the underlying grammar is individually designed for those distinct purposes. Most of the grammars known in the prior art are manually written in that sense that the rules constituting the grammar cover a huge set of phrases and various combinations of phrases that may appear within a dialogue.

In order to perform a mapping between a phrase and a semantic tag, the phrase or the combination of phrases has to match at least one of the rules of the manually written grammar. The generation of such a hand written grammar is an extreme time consuming and resource wasting process, since every possible combination of phrases or variations of a dialogue have to be explicitly taken into account by means of individual rules. Furthermore a manually created grammar is always subject to maintenance, because the underlying set of rules may not cover all types of dialogues and types of phrases that typically occur during operation of the automatic dialogue system.

In general, grammars for automatic dialogue systems are application related, which means that a distinct grammar is always designated to a distinct type of automatic dialogue system. Therefore, for each type of automatic dialogue system a special grammar has to be manually constructed. It is clear that such a generation of a multiplicity of different grammars represents a considerable cost factor which should be minimized.

In order to reduce a rather costly amount of manual efforts for generation, maintenance and adaptation of grammars, methods for an automatic generation of grammars or automatic learning of grammars have been introduced recently. An automatic construction of a grammar is typically based on a corpus of weekly annotated training sentences. Such a training corpus can for example be derived by logging the dialogue of an existing application. However, an automatic learning further requires a set of annotations indicating which phrases of the training corpus are assigned to which known tag. Typically, this annotation has to be performed manually but it is in general less time consuming than the generation of an entire grammar.

The paper "K Macherey, F. J. Och and H. Ney; Natural Language Understanding using Statistical Machine Translation', presented at the 7th European Conference on Speech Communication and Technology, Aalborg, Denmark, September 2001" which is also available from the URL "http://wasserstoff.informatik.rwth-aachen.de/Colleagues/och/eurospeech2001.ps" describes the automatic learning of a grammar.

In fact the document discloses an approach to natural language understanding, which is derived from the field of statistical machine translation. The problem of natural language understanding is described as a translation from source sentence to a formal language target sentence. This method therefore aims to reduce the employment of grammars in favour of a learning of dependencies between words and their meaning automatically. To this extent the mentioned method deals with a translational problem rather than with the automatic generation of a grammar.

In contrast to that, the US Patent application US 2003/0061024 A1 explicitly concentrates on the learning of a grammar. This method is based on determining sequences of terminals or of terminals and wild cards linked to non terminals of a grammar in a training corpus of sentences. After sequences of terminals or terminals and wild cards have been determined they are assigned to a non terminal or no non terminal by means of a classification procedure. This classification in turn uses an exchange procedure which is based on an exchange algorithm. The exchange algorithm guarantees an efficient optimization of a target function which takes account of all incorrect classifications and which is iteratively optimized in the classification of the sequences of terminals or of terminals and wild cards. Thereby the order of the non terminals in the training sentences does not have to be annotated manually since the target function uses only the information as to which sequences of terminals or of terminals and wild cards and which non terminals are present in the training sentences.

Furthermore, the exchange procedure guarantees an efficient (local) optimization of the target function since only a few operations are necessary for calculating the change in the target function upon the execution of an exchange.

The present invention aims to provide another method for mapping semantic tags to phrases and thereby providing the generation of a grammar for an automatic dialogue system.

The invention provides an automatic learning of semantically useful word phrases from weekly annotated corpus sentences. Thereby a probabilistic dependency between word phrases and semantic concepts or semantic tags is estimated. The probabilistic dependency describes the likelihood that a given phrase is mapped or assigned to a distinct semantic tag. In this context a phrase is used as a generic term for fragments of a sentence, a sequence of words or in the minimal case a single word.

The probabilistic dependency between phrases and tags is further denoted as mapping probability and its determination is based on the training corpus of sentences. Initially, the method has no information about the annotation between tags and phrases of the training corpus. In order to perform a calculation of the mapping probability a weak annotation between phrases and semantic tags must be somehow provided. Such a weak annotation can be realized for example by assigning a set of candidate semantic tags to a phrase. Alternatively an IEL (inclusion/exclusion list) can be used. An IEL represents a list that includes or excludes various semantic tags that can be mapped or must not map a phrase.

According to a preferred embodiment of the invention, for each phrase of the training corpus an entire set of mapping probabilities between the phrase and the corresponding set of candidate semantic tags is determined. In this way a probability that a given phrase is assigned to a semantic tag is calculated for each possible combination between the phrase and the entire set of candidate semantic tags which yields in an automatic learning or generation of a grammar.

According to a further preferred embodiment of the invention, a semantic tag is mapped to a phrase of the training corpus in accordance to the highest mapping probability of the set of mapping probabilities. This means that the mapping or assigning of a tag to a given phrase of the training corpus is determined by the highest probability of the set of mapping probabilities for the given phrase.

The method for mapping semantic tags to phrases makes therefore explicit use of the determination of mapping probabilities. Such a mapping probability can for example be determined from the given weak annotation between phrases and semantic tags of the training corpus. Generally, there exists a plurality of probabilistic means to generate such a mapping probability.

According to a further preferred embodiment of the invention, the statistical procedure, hence the calculation of the mapping probabilities, is performed by means of a expectation maximization (EM algorithm). EM algorithms are commonly known from forward backward training for Hidden Markov Models (HMM). A specific implementation of the EM algorithm for the calculation of mapping probabilities is given in the mathematical annex.

According to a further preferred embodiment of the invention, a grammar can be derived from the performed mappings between a candidate semantic tag and a phrase. Preferably the calculated and performed mappings are stored by some kind of storing means in order to keep the computational efforts on a low level. Finally, the derived grammar can be applied to new, unknown sentences.

The overall performance of the method of the invention can be enhanced when the EM algorithm is applied iteratively. In this case the result of an iteration of the EM algorithm is used as input for the next iteration. For example an estimated probability that a phrase is mapped to a tag is stored by some kind of storing means and can then be reused in a proceeding application of the EM algorithm. In a similar way the initial conditions in form of weak annotations between phrases and tags or in form of an IEL can be modified according to previously performed mapping procedures according to the EM algorithm.

In order to test the efficiency and reliability of an EM based algorithm for grammar learning, the EM based algorithm has been implemented by making use of a so called Boston Restaurant Guide corpus. Experiments based on this implementation demonstrate that an EM based procedure leads to better results than a procedure based on an exchange algorithm as illustrated in US Pat No. 2003/0061024 A1, especially when large training corpora are used. Furthermore, it has been demonstrated, that a repeated application of the EM based procedure leads to continuous improvements of the generated grammar. The tag error rate, which is defined as the ratio between the number of falsely mapped tags and the total number of tags, shows a monotone descent when described as a function over the number of iterations. The main improvements of the tag error rate are already reached after two or even one iteration.

In the following, preferred embodiments of the invention will be described in greater detail by making reference to the drawings in which:

Fig. 1 is illustrative of a flow chart for the mapping of phrases and tags by means of an EM based algorithm,
Fig. 2 shows a flow chart illustrating a dynamic programming construction of a table L which is a subroutine for the EM algorithm,
Fig. 3 is illustrative of a flow chart describing the implementation of the EM algorithm.

Figure 1 shows a flow chart for mapping of semantic tags to phrase based on the EM algorithm. In a first step 100 a phrase $\overline{w}$ is extracted from a training corpus sentence. In the following step 102 a step of mapping probabilities $p(k,w)$ for each tag k from a list of unordered tags κ.

Once a set of mapping probabilities has been calculated for the phrase $\overline{w}$, the highest probability of the set of mapping probabilities $p(k,w)$ is determined in the following step 104. In the next step 106 the mapping between the phrase $\overline{w}$ and a semantic tag k is performed. The phrase $\overline{w}$ is mapped to a single tag k according to the highest probability $p(k,w)$ of the set of mapping probabilities, which has been determined in step 104. In this way the mapping between a semantic tag k and a phrase $\overline{w}$ is performed by making use of a probabilistic estimation based on a training corpus. The probabilistic estimation determines the likelihood, that a semantic tag k is mapped to a phrase $\overline{w}$ within the training corpus. When the mapping has been performed in step 106 it is stored by some kind of storing means in step 108 in order to provide the performed mapping for a proceeding application of the algorithm. In this way, the procedure can be performed iteratively leading to a decrease of the tag error rate and thus to an enhancement of the reliability and efficiency of the entire grammar learning procedure.

The calculation of the mapping probability which is performed in step 102 is based on the EM algorithm, which is explicitly explained in the mathematical annex by making reference to figure 2 and figure 3.

The calculation of the mapping probability according to the EM algorithm is based on two additional probabilities denoted as $L(i,\kappa')$, and $R(i,\kappa')$, respectively, representing the probabilities for all permutations of an unordered tag sublist κ' of length $i-1$ over the left subsentence and the unordered complement tag sublist over the right subsentence of a training corpus sentence from position $i+1$.

Figure 2 is illustrative of a flow chart for calculating the probability $L(i,\kappa')$.

In a first step 200, the initial probability for $i=0$ is set to unity before in the next step 202, the index of the tag sublist i is initialized to $i=1$. In the following step 204, each sublist of length i is selected from the unordered tag sublist κ'. After selecting each sublist the calculation procedure continues with step 206, in which the probability $L(i,\kappa')=0$ for a permutation is set to zero. Then, in step 208 each tag k from the unordered sublist is selected in step 208, and successively provided to step 210, in which the permutation probability is calculated according to:

$$L(i,\kappa') = L(i,\kappa') + L(i-1,\kappa' \setminus \{k\}) \cdot p(k \mid \overline{w}_i).$$

After the calculation of $L(i,\kappa')$, in step 212, the index i is compared to the number of words in the phrase $\overline{W}$. If i is less or equal $|\overline{W}|$, the procedure returns to step 204 by incrementing index i by one. Otherwise, when i is larger than $|\overline{W}|$, the procedure for calculating the permutation probability ends with step 214.

Once the permutation probability has been calculated according to the procedure described in figure 2, an analog calculation is performed in order to obtain the permutation probability R for the complement sublist of the right subsentence.

Figure 3 finally illustrates the implementation of the EM algorithm for calculating a mapping probability $\tilde{p}(k,\overline{w})$ by making use of the above described permutation probabilities.

In the first step 300 for all tags k and phrases w the probability $p(k \mid \overline{w})$ is initialized by setting $\tilde{q}=0$ and setting $\tilde{q}(k,\overline{w})=0$, before in step 302 one of the training corpus sentences is selected. Since every sentence of the training corpus is taken into account for the grammar learning, the following step 304 has to be applied to all sentences of the training corpus.

After a sentence of the training corpus has been selected in step 302 it is further processed in step 304, in which the steps 306, 308, 310, and 312 are successively performed. In step 306, an unordered tag list κ' as well as an ordered phrase list $\overline{W}$ are selected. In the next step 308, the dynamic programming construction of the table L is performed as described in figure 2. After that, a similar procedure is performed with the reversed table R in step 310.

The calculated tables L and R as well as the initialized probabilities are further processed in step 312. Step 312 can be interpreted as a nested loop with an index $i=1,\ i \le |W|$. For each i, step 314 is performed initializing another loop for each of the unordered sublists κ of length $i-1$. For each unordered sublist the step 316 is performed selecting each tag $k \notin \kappa'$ and performing the following calculation in step 318:

$$\tilde{q}' = L(i-1,\kappa') \cdot p(k \mid \overline{w}_i) \cdot R(i+1,(\kappa \setminus \kappa' \setminus \{k\}),$$

where $\tilde{q}'$ is further processed in step 320 according to:

$$\tilde{q}(k,\overline{w}_i) = \tilde{q}(k,\overline{w}_i) + \tilde{q}' \text{ and } \tilde{q} = \tilde{q} + \tilde{q}'.$$

When the steps 318 and 320 have been executed for each tag $k \notin \kappa'$ in step 316, when step 316 has been performed for each unordered sublist of length $i-1$ in step 314, when step 314 has been performed for each index $i \le |\overline{W}|$ in step 312, and when finally the entire procedure given by step 312 has been performed for each sentence of the training corpus, then in step 322 the mapping probability is determined according to:

$$\tilde{p}(k,\overline{w}) = \tilde{q}(k,\overline{w}) / \tilde{q}\ \forall k,w.$$

Once the mapping probability has been determined, it is preferably stored by some kind of storing means. For the purpose of grammar learning and for mapping a tag to a given phrase all probabilities of all possible combinations of phrases and candidate semantic tags are calculated and stored. Finally, the mapping of a semantic tag to a given phrase is performed according to the maximum probability of all calculated probabilities for the given phrase.

Based on the plurality of performed mappings, the grammar is finally deduced and can be applied to other and hence unknown sentences that may occur in the framework of an automated dialog system.

Especially when the EM algorithm is repeatedly applied to a training corpus of sentences, the overall efficiency of the grammar learning procedure increases and the tag error rate decreases.

MATHEMATICAL ANNEX

According to a preferred embodiment of the invention, the mapping probability $\tilde{p}(k,\overline{w})$, that a given phrase $\overline{w}$ is mapped to a semantic tag k is calculated by means of an expectation maximization (EM) algorithm. The implementation and adaptation of a EM algorithm are described in this section.

Here, an approach which is similar to forward backward training of HMMs is followed. The general equation for EM based grammar learning is given by:

$$\tilde{p}(k,\overline{w}) = \frac{\sum_K p(K \mid W) \cdot N_K(k,\overline{w})}{\sum_K p(K \mid W) \sum_{\overline{w}',k'} N_K(k',\overline{w}')}, \qquad (1)$$

where W is a sequence of phrases, K is a tag sequence, $\overline{w}$ is a phrase, k

is a semantic tag, $N_K(k,\overline{w})$ is the occurrence that k and $\overline{w}$ occur together for a given W and K, and $p(K|W)$ gives the probability that a sequence of phrases W is mapped to a tag sequence K.

This approach assumes that the number of tags s equals the number of phrases. The numerator of equation (1):

$$\sum_K p(K \mid W) \cdot N_K(k,\overline{w})$$

adds for each tag sequence K the probability $p(K|W)$ as many times as the tag k is mapped to phrase $\overline{w}$ in this tag sequence. This may be rewritten as follows:

$$\begin{aligned}\sum_K p(K \mid W) \cdot N_K(k,\overline{w}) &= \sum_K \sum_i p(K \mid W) \cdot \delta(k_i,k) \cdot \delta(\overline{w}_i,\overline{w}) \\ &= \sum_{i:\overline{w}_i=\overline{w}} \underbrace{\sum_{K:k_i=k} p(K \mid W)}_{=p(k_i=k|W)}\end{aligned}$$

where $\delta(x,y)$ is the usual delta function

$$\delta(x,y) = \begin{cases} 1, x = y \\ 0, else \end{cases}$$

and $p(k_i = k \mid W)$ is the overall probability that the phrase $\overline{w}$ at position i in the phrase string W is mapped to tag k. Similarly, for the denominator of Eq.(1) the following holds:

$$\sum_K p(K \mid W) \cdot \sum_{k',\overline{w}'} N_K(k',\overline{w}') = \sum_{k',\overline{w}'} \sum_K p(K \mid W) \cdot N_K(k',\overline{w}')$$
$$= \sum_{i,k'} p(k_i = k' \mid W),$$

resulting into the estimation formula

$$\tilde{p}(k,\overline{w}) = \frac{\sum_{i:\overline{w}_i = \overline{w}} p(k_i = k \mid W)}{\sum_{i,k'} p(k_i = k' \mid W)}. \qquad (2)$$

For the estimation over the whole corpus, numerator and denominator must be separately computed und summed up for each corpus sentence.

The probability $p(k_i = k \mid W)$ that is central to Eq.(1) computes the probability of all tag sequences that have tag k for the phrase at position i. Before and after position i, all remaining permutations of tags are possible. If κ is the unordered list of tags and $\pi(\kappa)$ the set of all possible permutations over κ then

$$p(k_i = k \mid W)$$
$$= \sum_{K \in \pi(\kappa): k_i = k} p\ (K \mid W)$$
$$= \sum_{K \in \pi(\kappa): k_i = k} \left(\prod_{j=1}^{i-1} p(k_j \mid \overline{w}_j) \right) p(k \mid \overline{w}_i) \cdot \left(\prod_{j=i+1}^{s} p(k_j \mid \overline{w}_j) \right)$$
$$= \sum_{\kappa' \subseteq (\kappa \setminus \{k\}): |\kappa'| = i-1} \underbrace{\left(\sum_{\pi(\kappa')} \left(\prod_{j=1}^{i-1} p(k_j \mid \overline{w}_j) \right) \right)}_{=L(i-1,\kappa')} \cdot p(k \mid \overline{w}_i) \cdot \underbrace{\left(\sum_{\pi((\kappa \setminus \kappa') \setminus \{k\})} \left(\prod_{j=j+1}^{s} p(k_j \mid \overline{w}_j \right) \right)}_{R(i+1,\kappa \setminus \kappa') \setminus \{k\})}$$

$L(i-1,\kappa')$ is the probability for all permutations of the unordered tag sublist κ' of length i-1 over the left subsentence up to position i-1, and $R(i+1,(\kappa \setminus \kappa') \setminus \{k\})$ is the probability for all permutations of the unordered complement tag sublist $(\kappa \setminus \kappa') \setminus \{k\}$ of length $s-i$ over the right subsentence from position $i+1$. These values can be recursively computed:

$$\begin{aligned} L(i,\kappa') &= \sum_{K\in\pi(\kappa')} \prod_{j=1}^{i} p(k_j \mid \overline{w}_j) \\ &= \sum_{\kappa\in\kappa'} \sum_{K\in\pi(\kappa'):k_i=k} \prod_{j=1}^{i} p(k_j \mid \overline{w}_j) \\ &= \sum_{\kappa\in\kappa'} p(k \mid \overline{w}_i) \sum_{K\in\pi(\kappa'\setminus\{k\})} \prod_{j=1}^{i-1} p(k_j \mid \overline{w}_j) \\ &= \sum_{\kappa\in\kappa'} p(k \mid \overline{w}_i)\cdot L(i-1,\kappa'\setminus\{k\})\,. \qquad (3) \end{aligned}$$

Similarly,

$$R(i,\kappa') = \sum_{\kappa\in\kappa'} p(k \mid \overline{w}_i)\cdot R(i+1,\kappa'\setminus\{k\})\,. \qquad (4)$$

Storing and re-using the values $L(i,\kappa')$ and $R(i,\kappa')$ in Eqs. (3) and (4) reduces computational costs. For a given i, there are $\binom{|\kappa|}{i}$ unordered tag lists κ' and thus $\sum_{i=1}^{|\kappa|-1}\binom{|\kappa|}{i}\cdot i$ operations to perform to fully compute the table L (same holds for table R). However, no closed form or good estimation for this has been found, so it is not clear whether the computation is not efficient in the sense that it has a polynomial computing time.

The implementation of the EM algorithm is a direct consequence from the above mentioned expressions. The implementation is further described by Figs 2 and 3 for one iteration. There are just some notes about the implementation:

For technical reasons, each element of the unordered tag list κ gets a unique index in the range from 1 to $|\kappa|$. An unordered sublist κ' of length i is represented as an i- dimensional vector whose scalar elements are the indexes of the elements from κ that participate in κ'. This vector is incremented

$$\begin{pmatrix}1\\2\\.\\.\\.\\i-1\\i\end{pmatrix} \rightarrow \begin{pmatrix}1\\2\\.\\.\\.\\i-1\\i+1\end{pmatrix} \rightarrow\ldots\rightarrow \begin{pmatrix}1\\2\\.\\.\\.\\i-1\\|\kappa|\end{pmatrix} \rightarrow \begin{pmatrix}1\\2\\.\\.\\.\\i\\i+1\end{pmatrix} \rightarrow\ldots\rightarrow \begin{pmatrix}|\kappa|-i+1\\|\kappa|-i+2\\.\\.\\.\\|\kappa|-1\\|\kappa|\end{pmatrix}$$

to successively obtain all unordered sublists of length i.

The access to $L(i,\kappa')$ for some unordered sublist κ' of length i is realized by computing an index α with $L(i,\kappa') = L(\alpha)$ from the vector representation of κ':

$$\alpha = \sum_{j=1}^{i} 2^{a_j - 1},$$

where a_j is the j th element of the vector representation of κ'. The addition or removal of a tag to or from κ' is reflected in the index of the tag. The index β of the complement unordered list of tags needed for accessing $R(i,(\kappa \setminus \kappa') \setminus \{k\}) = R(\beta)$ is easily computed by

$$\beta = 2^{|\kappa|} - 1 - \alpha - 2^{a-1}.$$

For faster computation, there is a table whose j th entry contains the value 2^j.

The dynamic programming computation of the list R is performed by calling the subroutine that uses dynamic programming to compute the list L with a list of phrases $\overline{W}'$ whose phrase order is reversed, i.e. $\overline{w}_i' = \overline{w}_{s-i+1}$.

Sentences with an unequal number of tags and phrases are discarded.

The initial probabilities $p(k,\overline{w})$ are read in from a file and $p(\overline{w})$ is computed as marginal for $p(k \mid \overline{w})$. The file simply lists k, $\overline{w}$, and $p(k,\overline{w})$ in one ASCII line. The estimated probabilities $\tilde{p}(k,\overline{w})$ are written down in the same format and thus serve as input for the next iteration.

Figure 2 illustrates a flow chart for iteratively calculating the probability $L(i,\kappa')$ for all permutations of the unordered tag sublist κ' of length i over the left subsentence up to position i.

Initially, in step 200 the probability $L(0,\{\})$ is set to unity, before the index i is set to $i = 1$ in step 202.

In step 204, a loop starts and each unordered sublist κ' of length i is selected. In the proceeding step 206, the probability $L(i,\kappa') = 0$ for each selected unordered sublist is set to zero before in the next step 208 each tag k which is an element of the unordered sublist is selected. In step 210 finally, the probability $L(i,\kappa')$ is calculated according to:

$$L(i,\kappa') = L(i,\kappa') + L(i-1,\kappa' \setminus \{k\}) \cdot p(k \mid \overline{w}_i).$$

In step 212 it is checked whether the index i is smaller or equal the number of words in the phrase. If $i \le |\overline{W}|$ in step 212, then i is incremented by one, and the procedure returns to step 204. When in contrast $i > |\overline{W}|$, then the procedure stops in step 214.

The calculation of the probability for all permutations of the unordered complement tag sublist of the right subsentence from position $i+1$ is performed correspondingly.

Figure 3 is illustrative of a flow chart diagram for calculating a mapping probability $\tilde{p}(k,\overline{w})$ on the basis of the EM algorithm. In step 300 for all tags k and phrases w the probability $p(k \mid \overline{w})$ is initialized by setting $\tilde{q} = 0$ and setting $\tilde{q}(k,\overline{w}) = 0$, before in step 302 one of the training corpus sentences is selected. Since every sentence of the training corpus is taken into account for the grammar learning, the following step 304 has to be applied to all sentences of the training corpus.

After a sentence of the training corpus has been selected in step 302 it is further processed in step 304, in which the steps 306, 308, 310, and 312 are successively applied. In step 306, an unordered tag list κ as well as an ordered phrase list $\overline{W}$ are selected. In the next step 308, the dynamic programming construction of the table L is performed as described in figure 2. After that, a similar procedure is performed with the reversed table R in step 310.

The calculated tables as well as the initialized probabilities are further processed in step 312. Step 312 can be interpreted as a nested loop with an index $i = 1, i \le |W|$. For each i step 314 is performed initializing another loop for each of the unordered sublists κ' of length $i-1$. For each unordered sublist the step 316 is performed selecting each tag $k \notin \kappa'$ and performing the following calculation in step 318:

$$\tilde{q}' = L(i-1,\kappa') \cdot p(k \mid \overline{w}_i) \cdot R(i+1,(\kappa \setminus \kappa') \setminus \{k\}),$$

where $\tilde{q}'$ is further processed in step 320 according to:

$$\tilde{q}(k,\overline{w}_i) = \tilde{q}(k,\overline{w}_i) + \tilde{q}' \text{ and } \tilde{q} = \tilde{q} + \tilde{q}'.$$

When the steps 318 and 320 have been executed for each tag $k \notin \kappa'$ in step 316, when step 316 has been performed for each unordered sublist of length $i-1$ in step 314, when step 314 has been performed for each index $i \leq |\overline{W}|$ in step 312, and when finally the entire procedure given by step 312 has been performed for each sentence of the training corpus, then in step 322 the mapping probability is determined according to:

$$\tilde{p}(k,\overline{w}) = \tilde{q}(k,\overline{w})/\tilde{q}\ \forall k, w.$$